IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: John F. Wakerly

Application Serial No: 10/669,624

Filed: September 24, 2003

Confirmation No: 5626

Art Unit: 2446

Title: PARTITIONED PACKET PROCESSING IN A
                                MULTIPROCESSOR ENVIRONMENT

By email to benjamin.bruckart@uspto.gov

PROPOSED AMENDMENTS FOR DISCUSSION

Attached are proposed amendments based on suggestions made by Examiner Bruckart in a telephone call on June 24, 2009. Applicant has made amendments to Claims 1, 20, 29, 38, and 47, and canceled Claims 7, 8, 26, 34, and 43. The following amendments are for discussion purposes only and should not be placed in the permanent file of the Application.

IN THE CLAIMS

1.   **(Proposed Amendment)**   A system for packet processing, the system comprising:

a shared memory maintaining a plurality of code partitions, the code partitions together implementing a feature set for packet processing;

a plurality of processors each comprising a processor core and an instruction memory loaded with at least one of the code partitions from the shared memory, the processor core operable to execute the loaded code partition to perform processing of packets and to generate migration requests for transferring packet processing operations from the loaded code partition;

a context manager operable to receive a migration request from one of the loaded code partitions executing within one of the processor cores, the migration request comprising packet context information and identifying a target one of the code partitions, the context manager further operable, in response to the migration request, to identify an available one of the processors having the target code partition loaded, and to communicate the packet context to the available one of the processors; [[and]]

wherein the context manager maintains a plurality of queues each corresponding to one of the code partitions, the context manager further operable, in response to the migration request, to place migration data comprising the packet context information into the queue associated with the target code partition, to monitor the queue associated with the target code partition, and upon determining that one of the processors having the target code partition loaded is available for processing, to communicate the packet context information to the available one of the processors;

wherein an initial one of the code partitions includes instructions for initial packet processing that identify a plurality of processing functions and, for each of the processing functions, include a migration instruction associated with the processing function that indicates another one of the code partitions; and

wherein a selected one of the processors assigned the initial code partition is operable to receive packet context information associated with a received packet, the selected processor further operable, by executing the initial code partition, to identify characteristics of the received packet that correspond to one of the processing functions, to select the migration instruction associated with the identified processing function, and to generate a

migration request that comprises the received packet context information and targets the other one of the code partitions indicated by the selected migration instruction.

2.   **(Canceled)**

3.   **(Previously Presented)** The system of Claim 1, wherein the context manager is further operable to service each of the queues using a first in first out servicing schedule.

4.   **(Previously Presented)** The system of Claim 1, wherein the context manager is further operable to track an age for each entry in the queues and to service each of the queues based on the age for each of the entries.

5.   **(Original)** The system of Claim 4, wherein the age for each of the entries identifies a time when a packet corresponding to the entry was received by the system.

6.   **(Previously Presented)** The system of Claim 1, wherein prior to placing the migration data into the queue associated with the target code partition, the context manager is further operable to determine that the queue associated with the target code partition is empty and, in response, to bypass the queue by communicating the packet context information to the available one of the processors.

7.   **(Canceled)**

8.   **(Canceled)**

9.   **(Original)** The system of Claim 1, wherein the packet context information comprises a stack pointer that indicates a location in the shared memory.

10.   **(Original)** The system of Claim 1, wherein each of the processors further comprises registers and is further operable, when processing a received packet, to transfer values out of selected ones of the registers into a stack in the shared memory prior to transmitting a migration request for the packet to the context manager.

11.   **(Original)** The system of Claim 1, further comprising:

a first interconnect coupling the shared memory and the processors; and

a second interconnect coupling the processors and the context manager, wherein the second interconnect provides a dedicated link for transferring at least a portion of packet processing information between the code partitions operating on the processors.

12.   **(Original)** The system of Claim 1, wherein the context manager is further operable to assign some or all of the code partitions among the processors, to detect unbalanced operation that delays processing due to a selected one of the code partitions, and to reassign the code partitions such that the selected one of the code partitions is assigned to an increased number of the processors after the reassignment.

13.   **(Original)** The system of Claim 1, wherein the migration request further identifies one of a plurality of entry points within the targeted code partition.

14.   **(Original)** The system of Claim 13, wherein the migration request identifies the entry point using a program-counter offset from the beginning of the targeted code partition.

15.   **(Original)** The system of Claim 13, wherein the migration request identifies the entry point using an index to a table entry.

16.   **(Original)** The system of Claim 1, wherein at least one of the code partitions in the shared memory is not loaded in the instruction memory of any of the processors.

17.   **(Original)** The system of Claim 16, wherein the context manager is further operable to receive a migration request targeting one of the code partitions not loaded into one of the instruction memories and, in response, to initiate loading of the targeted one of the code partitions into the instruction memory of at least one of the processors.

18.   **(Original)** The system of Claim 1, wherein each of the code partitions comprises one or more pages of instructions, and wherein each of the instruction memories is further operable to load selected ones of the code partitions using a paging scheme.

19.   **(Original)** The system of Claim 1, wherein at least one of the processors is further operable to execute a plurality of processing threads, each of the processing threads operable to separately perform processing of packets using a loaded one of the code partitions.

20.   **(Proposed Amendment)** A context manager for handling migration of packet processing, the context manager comprising:

a processor; and

a computer-readable medium comprising a program of instructions, the program of instructions comprising:

an interface operable to couple to a system comprising a plurality of processors and a shared memory maintaining a plurality of code partitions, wherein the code partitions together implement a feature set for packet processing and wherein each of the code partitions is assigned as unloaded or is assigned to at least one of the processors;

a migration manager operable to receive a migration request from a selected one of the processors, the migration request comprising packet context information and identifying a target one of the code partitions, the migration manager further operable, in response to the migration request, to identify an available one of the processors having the target code partition assigned, and to communicate the packet context to the available one of the processors; [[and]]

wherein the migration manager maintains a plurality of queues each corresponding to one of the code partitions and is further operable, in response to the migration request, to place migration data comprising the packet context information into the queue associated with the target code partition, to monitor the queue associated with the target code partition, and upon determining that one of the processors having the target code partition loaded is available for processing, to communicate the packet context information to the available one of the processors;

wherein an initial one of the code partitions includes instructions for initial packet processing that identify a plurality of processing functions and, for each of the processing functions, include a migration instruction associated with the processing function that indicates another one of the code partitions; and

wherein a selected one of the processors assigned the initial code partition is operable to receive packet context information associated with a received packet, the selected processor further operable, by executing the initial code partition, to identify characteristics of the received packet that correspond to one of the processing functions, to select the migration instruction associated with the identified processing function, and to generate a

migration request that comprises the received packet context information and targets the other one of the code partitions indicated by the selected migration instruction.

21.   **(Canceled)**

22.   **(Previously Presented)**   The context manager of Claim 20, wherein the migration manager is further operable to service each of the queues using a first in first out servicing schedule.

23.   **(Previously Presented)**   The context manager of Claim 20, wherein the migration manager is further operable to track an age for each entry in the queues and to service each of the queues based on the age for each of the entries.

24.   **(Original)** The context manager of Claim 23, wherein the age for each of the entries identifies a time when a packet corresponding to the entry was received by the system.

25.   **(Previously Presented)** The context manager of Claim 20, wherein prior to placing the migration data into the queue associated with the target code partition, the migration manager is further operable to determine that the queue associated with the target code partition is empty and, in response, to bypass the queue by communicating the packet context information to the available one of the processors.

26.   **(Canceled)**

27.   **(Original)**   The context manager of Claim 20, wherein the packet context information comprises a stack pointer that indicates a location in a shared memory resource that is coupled to and accessible by each of the processors.

28.   **(Original)** The context manager of Claim 20, wherein the migration manager is further operable to detect unbalanced operation that delays processing due to a selected one of the code partitions and to reassign the code partitions such that the selected one of the code partitions is assigned to an increased number of the processors after the reassignment.

29.   **(Proposed Amendment)**   A method for handling migration of packet processing, the method comprising:

providing a system comprising a plurality of processors and a shared memory maintaining a plurality of code partitions, wherein the code partitions together implement a feature set for packet processing and wherein each of the code partitions is assigned as unloaded or is assigned to at least one of the processors;

receiving a migration request from a selected one of the processors, the migration request comprising packet context information and identifying a target one of the code partitions;

in response to the migration request, identifying an available one of the processors having the target code partition assigned;

communicating the packet context information to the available one of the processors;

maintaining a plurality of queues each corresponding to one of the code partitions;

in response to the migration request, placing migration data comprising the packet context information into the queue associated with the target code partition;

monitoring the queue associated with the target code partition; [[and]]

upon determining that one of the processors having the target code partition assigned is available for processing, communicating the packet context information to the available one of the processors;

wherein an initial one of the code partitions includes instructions for initial packet processing that identify a plurality of processing functions and, for each of the processing functions, include a migration instruction associated with the processing function that indicates another one of the code partitions; and

wherein a selected one of the processors assigned the initial code partition is operable to receive packet context information associated with a received packet, the selected processor further operable, by executing the initial code partition, to identify characteristics of the received packet that correspond to one of the processing functions, to select the migration instruction associated with the identified processing function, and to generate a migration request that comprises the received packet context information and targets the other one of the code partitions indicated by the selected migration instruction.

30.   **(Canceled)**

31.   **(Previously Presented)**   The method of Claim 29, further comprising servicing each of the queues using a first in first out servicing schedule.

32.   **(Previously Presented)** The method of Claim 29, further comprising:

tracking an age for each entry in each of the queues; and

servicing each of the queues based on the age for each of the entries.

33.   **(Previously Presented)** The method of Claim 29, further comprising, prior to placing the migration data into the queue associated with the target code partition, determining that the queue associated with the target code partition is empty and, in response, bypassing the queue by communicating the packet context information to the available one of the processors.

34.   **(Canceled)**

35.   **(Original)** The method of Claim 29, wherein the packet context comprises a stack pointer that indicates a location in a shared memory resource that is coupled to and accessible by each of the processors.

36.   **(Original)** The method of Claim 29, further comprising:

detecting unbalanced operation that delays processing due to a selected one of the code partitions;

determining assignments of the code partitions among the processors; and

reassigning the code partitions such that the selected one of the code partitions is assigned to an increased number of the processors after the reassignment.

37.   **(Original)** The method of Claim 29, further comprising receiving a migration request targeting one of the code partitions assigned as unloaded and, in response, loading the targeted one of the code partitions into the instruction memory of at least one of the processors.

38.   **(Proposed Amendment)** A computer-readable medium comprising logic for handling migration of packet processing, the logic operable when executed to perform the steps of:

detecting a system comprising a plurality of processors and a shared memory maintaining a plurality of code partitions, wherein the code partitions together implement a feature set for packet processing and wherein each of the code partitions is assigned as unloaded or is assigned to at least one of the processors;

receiving a migration request from a selected one of the processors, the migration request comprising packet context information and identifying a target one of the code partitions;

in response to the migration request, identifying an available one of the processors having the target code partition assigned;

communicating the packet context information to the available one of the processors;

maintaining a plurality of queues each corresponding to one of the code partitions;

in response to the migration request, placing migration data comprising the packet context information into the queue associated with the target code partition;

monitoring the queue associated with the target code partition; [[and]]

upon determining that one of the processors having the target code assigned is available for processing, communicating the packet context information to the available one of the processors;

wherein an initial one of the code partitions includes instructions for initial packet processing that identify a plurality of processing functions and, for each of the processing functions, include a migration instruction associated with the processing function that indicates another one of the code partitions; and

wherein a selected one of the processors assigned the initial code partition is operable to receive packet context information associated with a received packet, the selected processor further operable, by executing the initial code partition, to identify characteristics of the received packet that correspond to one of the processing functions, to select the migration instruction associated with the identified processing function, and to generate a migration request that comprises the received packet context information and targets the other one of the code partitions indicated by the selected migration instruction.

39.   **(Canceled)**

40.   **(Previously Presented)** The computer-readable medium of Claim 38, further operable to service each of the queues using a first in first out servicing schedule.

41.   **(Previously Presented)** The computer-readable medium of Claim 38, further operable when executed to perform the steps of:

tracking an age for each entry in each of the queues; and

servicing each of the queues based on the age for each of the entries.

42.   **(Previously Presented)** The computer-readable medium of Claim 38, further operable when executed to perform the steps of, prior to placing the migration data into the queue associated with the target code partition, determining that the queue associated with the target code partition is empty and, in response, bypassing the queue by communicating the packet context information to the available one of the processors.

43.   **(Canceled)**

44.   **(Previously Presented)**   The computer-readable medium of Claim 38, wherein the packet context comprises a stack pointer that indicates a location in a shared memory resource that is coupled to and accessible by each of the processors.

45.   **(Previously Presented)** The computer-readable medium of Claim 38, further operable when executed to perform the steps of:

detecting unbalanced operation that delays processing due to a selected one of the code partitions;

determining assignments of the code partitions among the processors; and

reassigning the code partitions such that the selected one of the code partitions is assigned to an increased number of the processors after the reassignment.

46.   **(Previously Presented)** The computer-readable medium of Claim 38, further operable to be executed on a selected one of the processors.

47.   **(Proposed Amendment)**   A system for packet processing, the system comprising:

memory means for maintaining a plurality of code partitions, the code partitions together implementing a feature set for packet processing;

a plurality of processing means each operable to access the memory means, to be loaded with one of the code partitions from the memory means, to execute the loaded code partition to perform processing of packets, and to generate migration requests for transferring packet processing operations from the loaded code partition; and

a context management means operable to receive a migration request from one of the loaded code partitions, the migration request including a packet context and identifying a target one of the code partitions, the context management means further operable, in response to the migration request, to identify an available one of the processing means having the target code partition loaded, and to communicate the packet context to the available one of the processing means; [[and]]

wherein the context manager means maintains a plurality of queues each corresponding to one of the code partitions, the context manager means further operable, in response to the migration request, to place migration data comprising the packet context information into the queue associated with the target code partition, to monitor the queue associated with the target code partition, and upon determining that one of the processing means having the target code partition loaded is available for processing, to communicate the packet context information to the available one of the processing means;

wherein an initial one of the code partitions includes instructions for initial packet processing that identify a plurality of processing functions and, for each of the processing functions, include a migration instruction associated with the processing function that indicates another one of the code partitions; and

wherein a selected one of the processors assigned the initial code partition is operable to receive packet context information associated with a received packet, the selected processor further operable, by executing the initial code partition, to identify characteristics of the received packet that correspond to one of the processing functions, to select the migration instruction associated with the identified processing function, and to generate a migration request that comprises the received packet context information and targets the other one of the code partitions indicated by the selected migration instruction.